L #4

* Loop: Repeating a set of instructions certain # of times.

Loop → ① Register ⟶ Counter
Loop → ② DJNZ : Decrement and jump if not zero

DJNZ Reg, Label

↳ Reg = Reg - 1 Then if Reg ≠ 0 ⟶ jump to Label

---

* write a program to add 5 to A 8 times

```
        ORG   0000H
        MOV   A, #0      ←  A=0 ⟷ CLR A
        MOV   R2, #8     ←  R2=8
LOOP:   ADD   A, #5      ←┐
        DJNZ  R2, LOOP   ──┘ 8
        SJMP  $
        END
```

what are the content of R2 and A at the end ?

R2 ↓ 0

A ↓ 5×8 = 40D

CLR ⟨ A / bit → CLR A ✓

CLR R3 X

CLR C ✓

---

what is the maximum number of times the loop can be repeated using 8-bit register? 256

→ 255

```
      MOV  R2, #0
Loop: ADD  A, #5
      DJNZ R2, Loop
```

256

- Write a program to load A with 55H then complement

A 1000 time.

- 50 X 20 ✓
- 100 X 10 ✓
- 25 X 40 ✓
- 250 X 4 ✓
- 500 X 2 X

```
        ORG   0000H
         MOV   A, #55H ←
         MOV   R3, #4
NEXT:    MOV   R4, #250
Loop:    CPL   A
         DJNZ  R4, Loop
         DJNZ  R3, NEXT
         SJMP  $
         END
```

250 x 4 = 1000

A = 0101 0101 → 55H ← even

1's

A = 1010 1010 → AAH ← odd

A A

At the end A = 55H

JUMP

conditional jump

(relative address)

unconditional jump

SJMP (relative address)

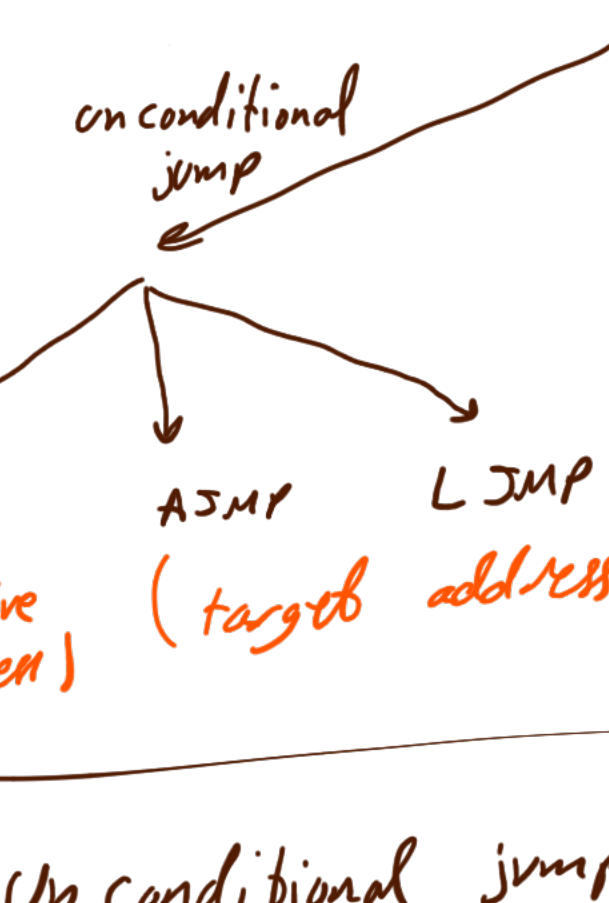
AJMP (target address)

LJMP

(I) Unconditional jump

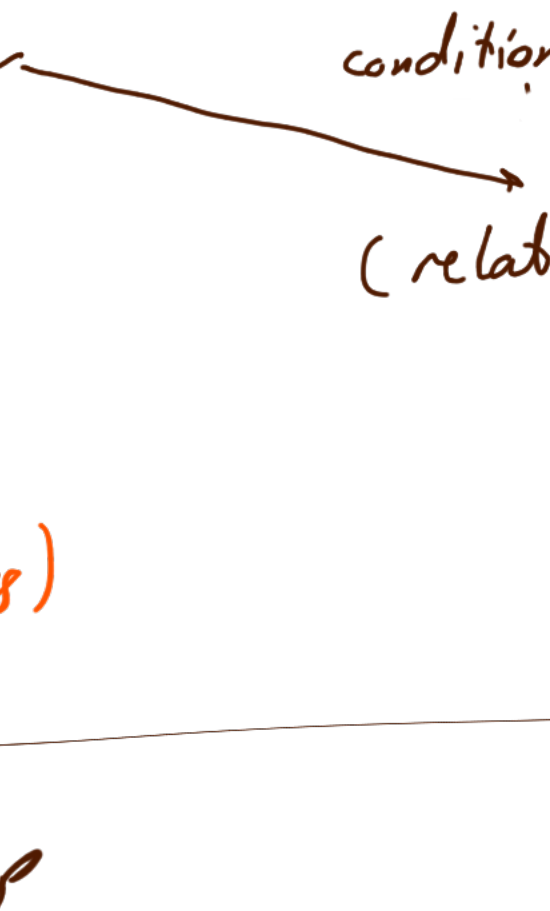
SJMP

short jump



AJMP

Absolute jump



LJMP

Long jump

① SJMP :

It is 2 byte instruction
- The first byte is op code (80H)
- second byte is called relative address
  - 1 byte (8-bit)

The range of relative address is 00H ⟶ FFH is divided into forward +127 byte and backward -128 byte relative to the PC.

② LJMP : long jump

It is 3 byte instruction
- The first byte is op code
- second and third bytes are called target address

Target address = 2 byte (16 bit)

Range → 0000H ⟶ FFFFH

Can jump up to $2^{16} = 2^6 \times 2^{10} = 64kB$

SJMP is faster than LJMP ? X

↓ 2 cycles

LJMP → 2 cycles

(II) Conditional jump : all conditional jump are relative address

① JZ label ; jump to the label if A=0

② JNZ " ; " " " " " A≠0

③ JC " ; " " " " if cy = 1

④ JNC " ; " " " " if cy = 0

⑤ JB Bit, label ; jump to the label if Bit = 1

⑥ JNB bit, label ; " " " " if Bit = 0

⑦ JBC bit, label ; " " " " if Bit = 1 then clear the bit.

⑧ DJNZ

⑨ CJNE : compare and jump if not equal.

* write a program to check the content of R7

if R7 = 0 ⟶ save 'Y' in R1

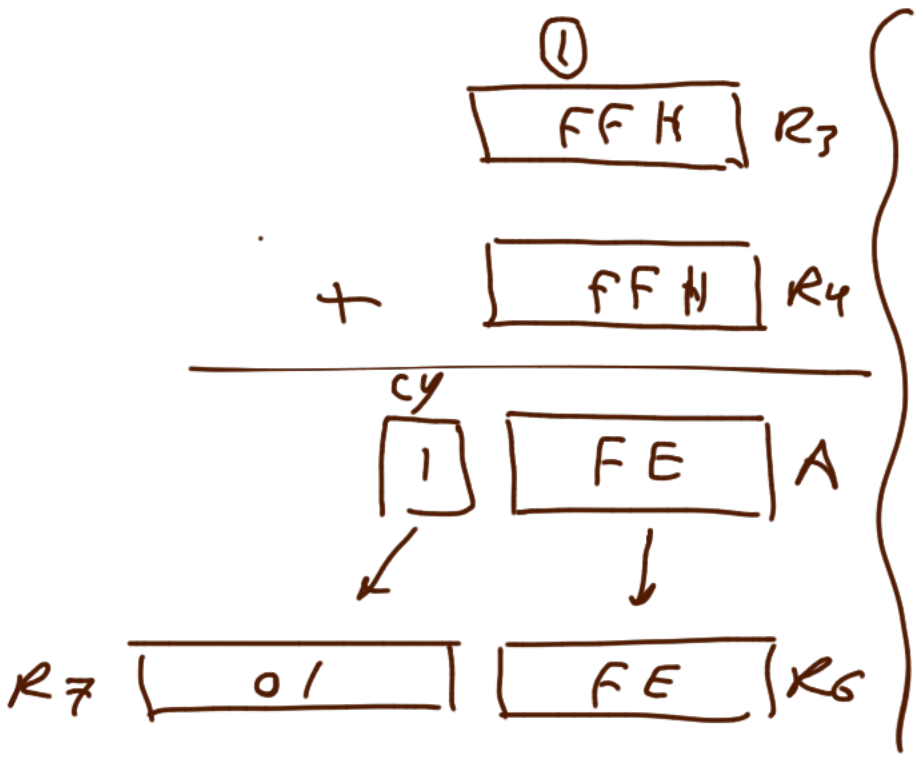
otherwise (R7 ≠ 0) ⟶ save 'N' in R1

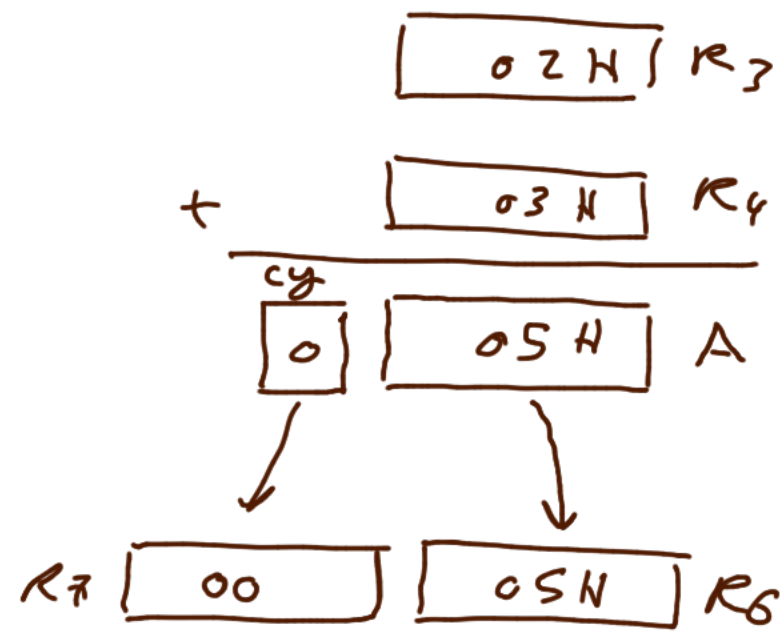
```
      MOV   A, R7
      JZ    NEXT
      MOV   R1, #'N'
      SJMP  HERE
NEXT: MOV   R1, #'Y'
HERE: SJMP  HERE
      END
```

R7=0

---

write a program to add the content of R3 and R4. Save result in R6 (low byte) and R7 (high byte).





```
ORG 0000H
MOV  R7, #00   <-
MOV  A, R3
ADD  A, R4
JNC  NEXT
INC  R7
NEXT: MOV  R6, A
SJMP $
```

cy = 0